UML Modeling Mechanism Based on Layered Structure of Abstraction*

Wu Yue Luo Wuman

(College of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract This paper introduces UML(Unified Modeling Language) modeling method. Based on the experience of traditional software development method and the analysis of traditional three layered structure of abstraction in UML, two ideas of abstraction and decomposition are brought forward as the kernel modeling ideas of UML to construct the abstraction modeling mechanism of UML that generalizes the process of computer system into the process of picking up five cells: the function cell. the structure cell, the service cell, the implementation cell and the integration cell. The purpose is to make use of the whole semantics of UML effectively and to get over the limitation of traditional OO method and eventually to achieve the perfectly transition from requirement to analysis and virtual separation of function and implementation.

Key words unified modeling language; abstraction; decomposition; layered modeling; cell

1 Some Concepts of UML

1.1 Use case

An use case is a coherent unit of functionality provided by a system or class as manifested by sequences of messages exchanged among the system and one or more outside interactors (called actors) together with actions performed by the system.

1.2 Class

A class is the descriptor for a set of objects with similar structure, behavior, and relationships. A class may use a set of interfaces to specify collections of operations it provides to its environment. Class may be abstract and executable.

1.3 Interface

An interface is a declaration of a collection of operations that may be used for defining a service offered by an instance of class (object). An interface serves to name a collection of operations and specify their signatures and protocols. An interface offers no implementation (method) for any of its operations.

1.4 Package

A package is a group of model elements such as classes. A package may contain both subordinate packages and ordinary model elements. Some packages may be subsystems or models. The entire system description can be thought of as a single high-level subsystem package with anything else in it.

1.5 Component

A component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces. A component may be one or more class.

Received on July 28, 2000

* The project supported by the Academic Foundation of Nuclear Physics of China

2　Two Modeling Ideas Brought Forward

Worldwide, there is an insatiable demand for software. On one hand, ever increasing demands on software for more of everything——functionality, flexibility, reusability, robustness——seem to ensure the software complexity keep pushing the limits of both tools and human intellect; on the other hand, it becomes more and more difficult to understand and express the large-grained behavior patterns that will be jointly achieved by the components of a system while the system is running[1,2]. So constructing perfect models for object system is the key task of the whole system design. For these reasons, it is urgent to find some guidance for modeling to simplify the whole design process and enhance the efficiency of software development.

Based on the experiences of engineering practices, we bring forward two kernel modeling ideas: abstraction and decomposition. They are also two basic thoughts of human when human recognize the world and manage the complexity of the problems that they are trying to resolve.

What is abstraction? When faced with a problem, we often want to know its common attributes while ignore the branches and knots. We call this method abstraction. In the field of software engineering, abstraction has two meanings. The first is that when analyzing the complex systems, designers distinguish the essential and nonessential factors and get rid of the nonessential things in order to get the essence of the problem. By doing so, designers can describe the structure of the object systems exactly and transform the structureless systems to the structural systems. Another meaning of abstraction is that during the process of modularization, designers generalize the functions of the lower layer modular to the higher layer modular. By doing so, mechanism of higher layer abstraction can be formed. Abstraction can be decomposed into two layers. The higher layer is called specification, which specifies the abstraction“what to do”. The lower layer is called realization, which resolves the problem of abstraction “how to do”. Because of the application of abstraction, software designers are able to consider problems in different layers of abstraction during different phases of development.

Decomposition is based on the strategy of divide and rule. It is used to resolve the design complexity of the object system as a whole. Decomposition is embodied as “modularization and information concealment” during the software development. From the view of technology, decomposition is the approach of the concretion of abstraction, for example, the idea of refinement step by step. Reverse to decomposition, composition is an important approach of software implementation after decomposition.

3　Three Layered Structure of Abstraction in UML

Fig.1 shows the traditional three layered structure of abstraction in UML[3]. We use the ideas of abstraction and decomposition to analyze it.

3.1　Conception

During this layer, all the model elements picked up from object system are conceptions in application domain. These conceptions are provided independent of the software and programming languages that realize them.

3.2　Specification

During this layer, all the concepts are evolved into the units that interact with each other to achieve an explicit function, which is the decomposition of the upper layer and the abstraction of the lower layer because it only describes the interfaces provided by function units while

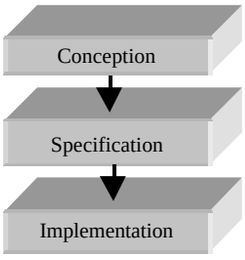

Fig.1　Three layered structure of abstraction

concealing the realization of them.

3.3 Implementation

This layer describes the realization of the function units. The models of this layer perhaps are more familiar to most of people. But in fact, the models of the specification layer are more in favor of the understanding and communication with each other between the developers.

4 Abstraction Modeling Mechanism of UML

Based on the three layered structure of abstraction, we bring forward the abstraction modeling mechanism of UML, shown in Fig.2. In this modeling mechanism, we focus on the key messages dominating the modeling process and abstract them to five different cells: the function cell, the structure cell, the service cell, the implementation cell and the integration cell, which represent five kernel stages at different abstraction layers. The cells born in former stages are the abstraction of the cells born in later stages[4].

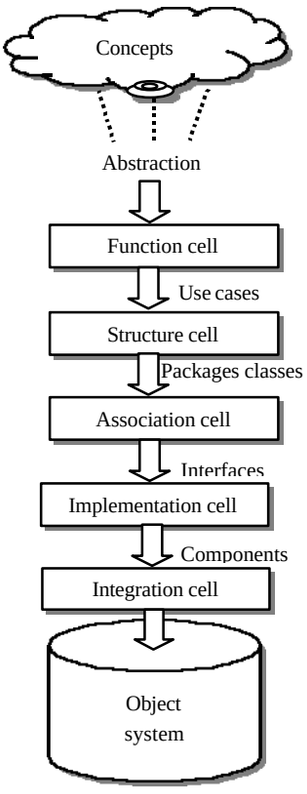

Fig. 2 Abstraction modeling mechanism of UML

4.1 Function Cell

People′s acquaintance of the application domain begins with its functions in it. Comprehending the functions aids in application domain understanding. So picking up the function cells will drive the key design decisions from the application domain and modeling, for it is the exordium of the whole modeling process. The function cell belongs to the conception layer of the abstraction structure in UML and is embodied as use cases[5]. An use case may be larger or smaller as long as it provides the integrated description of one user target. Modeling use cases is the lifelike recurrence of people′s mental impression in application domain.

Use cases have a strong effect on further design decisions. To find perfect use cases, designers rely on experience, iteration, and heuristics. So when picking up use cases from the system, we should think over: which functions should be provided by the system according to the demands of the actors[3], how many kinds of information should be read, created, deleted, modified, or stored by the actors, how many types of inputs and outputs are needed by systems, and where do the inputs come and outputs go.

Defining function cells is an advanced requirement analysis that segues seamlessly into high-level design for other kinds of cells are designed and added later to interact with each other to realize them.

4.2 Structure Cell

After defining the function cells and modeling for them, we may consider selecting structure cells used to interact with each other to accomplish a specific function described by the function cell. Structure cells are the units forming the whole software. A well selection of them is the foundation of the extensibility of the system. Structure cell also belongs to the conception layer and is embodied as classes and objects[5].

Classes and objects are the basic elements in object oriented design. The entities in the application domain map into classes and their associations in UML. Classes at this level serve as a vocabulary of essential design concepts and act as a basis for doing more detailed class-based design afterwards. Classes are useful as abstract concepts to model many things, not just class hierarchies in object oriented implementation. So at this level, they must be quite distant from the final implementation, thus aid in application domain understanding. Here give some guidelines for the selection of the classes at this level: only documenting the static entities of the application domain, using inheritance sparingly or downplaying behavior by ignoring methods.

4.3 Service Cell

Up to now the information identified by the cells we defined are all static characters in application domain. For example, we name the elevator and motor as the classes in the elevator controlling system, but ignore the problems such as which floor the elevator will go and the way the motor works. The purpose of picking up the service cells is to find the services provided by structure cells and the associations between them when they work. In one word, establish the foundation for constructing the dynamic associations between structure cells.

As described above, the specification layer is the best entrance for understanding the UML models. The only way to realize the structure cell is to know the services it provides. In UML, service cells are encapsulated as the interfaces in structure cells[5]. Because in object-oriented systems, each entity and services it needs or provides are mapped into a class and its interface. Each class provides and uses services through its interface, concretely speaking, through the operations defined in interfaces. We refine the classes one level up and apart the interfaces from the class itself and transform the classes from the conception level to the specification level. So the key work at this level is to decompose and refine the structure cells to specify the services they provide and need while concealing the realization details.

4.4 Implementation Cell

Up to now we have specified the framework of the whole system and come to the implementation abstraction layer. Now it is time to centralize our attention to define the implementation cells, in other words, to implement the method of each operation in the service cells[5].

In UML, the realization of the method is to design corresponding arithmetic and use specific programming language to implement it. As the result, the components of the software′ are made and the main work of software design is completed.

We can also pick up existing components to realize services cells. UML not only is based on OO but also supports the software reuse. Along with the accumulating of components, making use of the existing resources in reason can avoid the process from top to toe.

4.5 Integration Cell

Integration cell focuses on the realization of the whole system. We know that nowadays systems are often part software and part hardware and distributed over communication networks. System integration has become more and more important[6] .

The task of system integration is to map each kind of modeling element——class, object, operation, method etc, which are defined in function cell, structure cell, service cell and implementation cell to the physical computer nodes. According to the system, network topology structure integrates the different computers and different system operations. So the main work to do at this layer is to research the physical topology structure of system hardware and the software running on them. We call these information

related to system integration as integration cells. The integration cells in UML are embodied as nodes and their connections, components and their interfaces, objects.

5 Conclusion

The motivation we construct the layered abstraction modeling mechanism is to provide a thread for the research in UML modeling. Much more work should be done to refine the work of the selection five kinds of cells and the definitions mapped to UML, as well as combining the object oriented technology and software reuse technology perfectly to maximatily enhance the quality of our products.

Reference

1 Buhr R J A, Casselman R S. Use case maps for object-oriented systems. New York: Prentice-Hall International Inc,1999

2 Zhang Fengli, Ge Xiaofeng, Lu Xianliang. Design and implement of comprehensive query system based on data ware house. Journal of University of Electronic Science and Technology of China, 1999, 28(2):207~210 [**张凤荔**, **葛晓峰**, **卢显良**. **基于数据仓库的综合查询系统的设计和实现**. **电子科技大学学报**, 1999, 28(2):207~210]

3 Kruchten Philippe. Modeling component systems with the unified modeling language. Rational Software Corp, 1998

4 Mei Denghua. A object-oriented multicomputer software system reliability model. Journal of University of Electronic Science and Technology of China, 1999, 28(2):191~194 [**梅登华**. **面向客户多机系统的软件可靠性模型**. **电子科技大学学报**, 1999, 28(2):191~194]

5 UML Semantics Guide, version 1.1. 1997, http://www.rational.com/uml

6 Wu Yue, Yu Shui, Fu Yan, *et al.* On Internet database accessing technology. Journal of University of Electronic Science and Technology of China, 2001, 30(1):58~61 [**吴 跃**, **余 水**, **傅 彦**,**等**. Internet **数据库访问技术**. **电子科技大学学报**, 2001, 30(1):58~61]

统一建模语言 UML 分层抽象建模机制*

吴 跃** **罗吴蔓**

(**电子科技大学计算机科学与工程学院 成都** 610054)

【摘要】在总结传统软件开发方法经验的基础上，严格遵守软件开发方法原则，从 UML **建模思想出发，对传统** UML **三层抽象建模结构进行了分析，并引入“抽象”和“分解”作为** UML **建模的核心思想；提出了** UML **分层抽象建模机制的构想，将计算机系统开发过程概括为对功能信元、结构信元、关联信元、实现信元和集成信元的提取过程，并以此作为** UML **建模的指导。有效了利用** UML **完整语义定义，克服了传统面向对象开发方法的缺陷，实现了建模过程从需求到分析的过渡以及功能和实现事实上的分离。**

关 键 词 统一建模语言； 抽象； 分解； 分层建模； 信元

中图分类号 TP311

2000**年**7**月**28**日收稿**

* **中国核物理研究院基金资助项目**

** **男** 42**岁** **博士** **副教授**